DelDOT CAMS

Data Archival Design and Process

Version 2.1

TransCore, Inc.
8158 Adams Drive
Hummelstown, PA 17036

March 2019

Project Number 1313406, Contract Number 1601
Document Number 1313406-10

.

PAGE INTENTIONALLY LEFT BLANK

Revision History

VERSION #	DATE	SECTION #	CHANGE DESCRIPTION
Version 1.0	January 2011	All	Eric Strickland and Satish Chundru
Version 2.0	July 2014	All	Steve Merkel
Version 2.1	March 2019	All	Updated to show 24 months retention and indicate no archive and purge of summary data.

CONTRIBUTOR	NAME	DATE
ORIGINATOR(S)	Eric Strickland, Satish Chundru and Steve Merkel	July 2014
REVIEWER(S)	Satish Chundru	July 2014
AUTHORIZED BY TRANSCORE	Satish Chundru	July 2014
ORGANIZATIONAL APPROVALS		
DELDOT		
CONSULTANT		

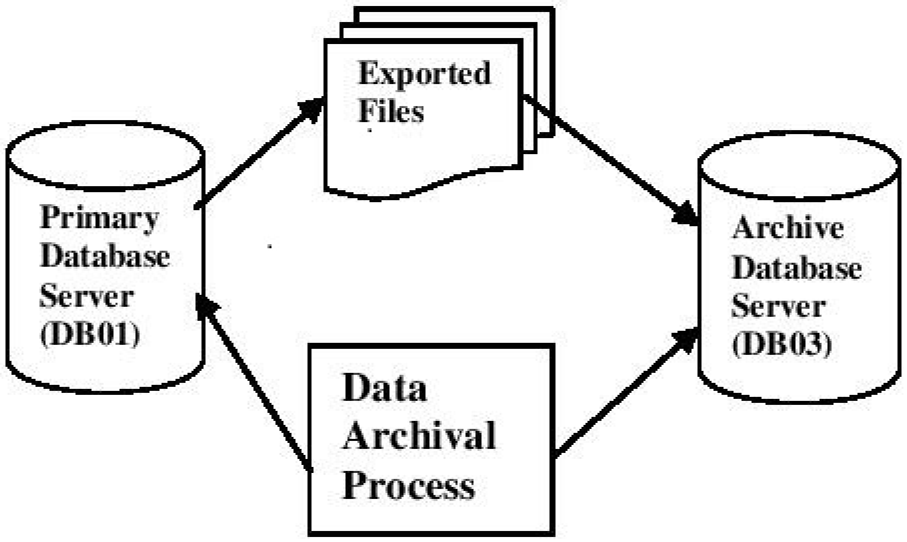
PAGE INTENTIONALLY LEFT BLANK

Contents

PAGE INTENTIONALLY LEFT BLANK

1 Data Archival Design

The data keeps accumulating in the production databases from the lanes, account process and report summaries. The data retention is at least 24 months for detail transactions and at least 5 years for summary transactions. The oldest month of data will be archived into the archive server. This design is very simple and elegant in that a table can be added or removed easily from the archival process through configuration tables. We can keep the transactions a configured number of months based on its type. The entire process is designed through the stored procedures, tables, views and SQL Server jobs. When the jobs runs once a month, it will identify the oldest month of data to be archived, it then exports that month's data to text files, purges that month's data from the production database and then imports that data into the Archive server. This way the purged data is being stored in the Archive server for later access for any research purposes. Here is a simple diagram which depicts the design of this process:


Exported Files
Primary Database Server (DB01)
Archive Database Server (DB03)
Data Archival Process

1.1 Purge/Archive Requirements

DelDOT Requirements for archiving/purging of transactions are dependent on transaction type. Here are the requirements:

1. ETC transactions (detail level transactions) shall be retained for a rolling eighteen (24) months and then archived to permanent long-term storage. These transactions will then be imported into the Archive server.

2. Summarized data shall be retained on the production server for at least five (5) years, and then archived to permanent long-term storage. These transactions will then be imported into the Archive server. The current procedure is to not archive and purge the summarized data. The

configuration –as described below--allows for archive and purge of Summary data, but this portion of the process is not currently being used.

Meeting these requirements requires two main part of the Archive process. The first part (Archive Export) runs on the production database server and it carries out the processing necessary to export and purge the database records from the production system. The second part of the process (Archive Load) runs on the Archive Server and does the processing required to take the data exported in the first part and load the into the appropriate tables in the Archive database

1.2 Purge/Archive Scope

Not all tables in the AMS and IAG databases will be subject to the Archive process. No account related data will be affected. In general, the only data to be purged from the online production databases (and stored in the Archive database), will be transactional data that can be later analyzed as needed in the Archive database.

The following table lists all the production database tables that will be affected by this process.

[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]

[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]

1.3 Archive Export Design

To meet the Purge/Archive requirements, the Archive Export process will be run once a month and will export and purge data that is older than the configured retention period.

There are 2 different transaction types of data can be exported and purged. Each has a separately configured retention period:

- ETC Transactions
- Summary Transactions (not currently configured)

For the ETC and Summary transactions, the value for configuration parameter is set to 1 (Archival) by default. In other words, these transactions are going to be exported to text files before the purge.

1.3.1 Export of ETC and Summary Transactions

The process of copying and deleting transactions is same for both ETC transactions and Summary transactions. The configuration table will have three important columns called “MonthsToKeep”, “ProceedWithCopy”, and “ProceedWithPurge”.

The field "MonthsToKeep" is going to be populated based on the number of months of transactions that should be retained in the database. The process will then compute the cutoff date. It will then archive all the data from the tables based on this date.

There are two steps:

- It will extract all the transactions and write them to the text files.
- It will then purge those transactions

1.3.2 Archive Export Tables

The Archive Export process utilizes a set of 4 tables. Two of the tables hold configuration information that drive the processing of the Archive Export, and the other two tables are for logging the progress of the process. There is a set of these four tables in both the AMS and the IAG production databases. The tables are created in the 'Archive' schema of each database.

1.3.2.1 Archive Export Configuration Tables

The two configuration tables are Archive.tbLkpArchiveConfig and Archive.tbLkpArchiveTableName.

1.3.2.1.1 [redacted]ArchiveConfig

This table stores all the configuration values used by the Archive Export process. This table will have two rows; one for ETC Transactions and one for Summary Transactions

The "MonthsToKeep" field should be populated with the number of months of transactions that should be retained in the database. "ProceedWithCopy" and "ProceedWithPurge" act as switches that control the process of Archival and Purge. The location of archive files should be specified in "DestinationFileLocation".

Below is the Schema information for this configuration table with descriptions of each column in the table and any noted configuration values.

COLUMN NAME	**DESCRIPTION**
ArchiveDataTypeID	This field indicated the type of transaction. 1 = ETC Transactions. 2 = Summary Transaction
ArchiveDataType	This field indicates a description of the record type.
ConfigurationID	This field indicated the type of configuration for the archival process. 1 = configuration record for archiving. 2 = configuration record for purging
MonthsToKeep	This field is to be populated with the number of months of transactions that should be retained in the database before archiving or purge. Example MonthsToKeep = 12 for the Archive record will archive data older than 12

	months from the last archive date.
DaysToKeep	The value of this field is computed based on the MonthsToKeep when the job runs.
DestinationLocation	This is the file location where the exported files will live on the server.
PreviousCutOffDate	This field is used to hold the previous Archive cutoff date. Used to make sure the Job does not run if the current date is not beyond the previous cutoff date.
CurrentCutOffDate	This field is used to contain the current Archive cutoff date during the job processing. When completed it becomes null and is moved to the PreviousCuttOffDate
ProceedWithCopy	This field act as switches that control the process of Archival
ProceedWithPurge	This field act as switches that control the process of Purge

1.3.2.1.2 Archive.tbLkpArchiveTableName

This table contains the names of all tables that are processed by the Archive Export process. Below is the Schema information for this configuration table with descriptions of each column in the table and any noted configuration values.

Column Name	DESCRIPTION
ArchiveDataTypeID	This field indicated the type of transaction. 1 = ETC (Detail) Transactions. 2 = Summary Transaction
ArchiveOrder	This field is used for the order to copy or purge. CAMS is a relational database and data is required to be deleted in a specific order for related data.
ArchiveSourceName	This field contains the source table name for archiving or purging used for. Sometimes this could also be a view. Example: tbETCTripTrxn
TableName	This field contains the table name for archiving the data from. Example: tbETCTripTrxn
KeyFieldName	This field contains the key field date information used to select data for archiving or purging. Examples: IYYYYMM, or dtUpdTime
KeyFieldFormat	This field defines the format of the KeyFieldName

	Example: 'YYYY-MM-DD'
PurgeJoinFieldName	This field contains the value for the type of "**KeyFieldName**" we used to join the view with the table. Example: iYYYYMM
PurgeDependencyName	This field will either list the name of the 'ArchiveSourceName' if it is a view, or 'Self' if there is no view involved
Active	The value of 1 indicates it this table is participating in the Archive Export process. The value of 0 indicates it is not.

1.3.3 Archive Export Log Tables

There are 2 log tables which keep track of the database Archive Export job runs:

1.3.3.1 [redacted]ArchiveSummary

When the Archival process starts, it inserts a record into this table. The value for "DaysToKeep" gets calculated based on the on the value of "MonthsToKeep" specified. "CutOffDate" is calculated based on the value stored in "DaysToKeep" field in Archive.tbLkpArchiveConfig table. Any transaction that is earlier than this "CutOffDate" is archived and purged. The "CopyState" and "PurgeState" are set to 0 at the beginning of the process. The "CopyState" gets set to 1 only when all the tables have been copied successfully. Similarly, the "PurgeState" gets set to 1 only when all the tables have been deleted successfully.

1.3.3.2 [redacted]ArchiveDetail

This table contains the logs for all the individual tables that belong to a particular set of the Archive Export process. All pertinent table names are retrieved from "Archive.tbLkpArchiveTablename" and based on the "CufOffDate", the "StartKeyValue", "EndKeyValue" and "TableTransCount" are obtained from individual tables and inserted into this table. The "CopyState" and "PurgeState" fields are set to 0. Transactions are copied from the tables using BCP process. After copying individual tables, the copied transactions count ("CopyTransCount") is checked against the table transactions count ("TableTransCount") and if there is a match, the "CopyState" is set to 1. In case of a problem during the Archival process, restarting the process would skip all the tables that have been successfully copied and start with the first one that is yet to be copied for a specific batch (this is determined by the max record in "tbArchiveLog" that has "CopyState" set to 0). Similarly, the purge process deletes the transactions that have been successfully copied. The deleted transactions count is checked against the table transactions count and if there is match, the "PurgeState" is set to 1. In case of problems during the purge process, restarting the process would continue with the table that has a PurgeState of 0.

1.3.3.3 Archive Export Stored Procedures

There are 4 different stored procedures used to carry out this process and they are as follows:

1. [redacted] copies detail transactions from tables into text files.
2. **A**[redacted]deletes all the detail transactions that were copied successfully

3. [redacted] copies summary transactions from tables into text files.
4. [redacted] deletes all the summary transactions that were copied successfully

1.3.3.4 Archive Export Job

We have one job called DatabaseArchive_Export_Job. This job is scheduled to call all 4 of the stored procedures in 4 different steps.

At the end it calls the MOMS stored procedure dbo.uspEquipFailureInsert in case of any step failed.

1.4 Archive Load Design

The data that is exported from the production databases by the Archive Export process is then loaded into the Archive Databases on the Archive Server. The Archive Load process is designed to accomplish this data loading. As a one-time process to prepare the Archive Databases for the loading, two modifications are made. First, any table that is to receive data via the Archive Load process must have the identity property removed. This ensures that the loaded data will retain the same primary key values that it had in the production system. Secondly, any foreign key constraints that would prevent the intended data loading are dropped in the Archive Databases.

1.4.1 Archive Load Components

The Archive Load process runs on the Archive Server. The process has the following components that will be described in detail below:

1. A database called ‘ArchiveStage’ that holds the database objects utilized by the Archive Load process.
2. A table in the ArchiveStage database called ‘Archive.tbArchiveFiles’ that tracks and logs the export files to be loaded, and the progress of loading each file.
3. A set of tables in the ArchiveStage database that are used to first stage the data from the export files in the ArchiveStage database.
4. A set of 5 stored procedures in the ArchiveStage database that are used to perform the Archive Load process.
5. A SQL Agent job in the Archive Server instance named ‘Load_Archive’ that calls the stored procedures that perform the Archive Load processing
6. Final destination-- Archive databases that will contain the loaded data for long term retention and analysis.

1.4.2 Database ‘ArchiveStage’

The ArchiveStage database contains the database objects that are utilized by the Archive Load process. The database has three Schemas in addition to the default ‘dbo’. They are: Archive, AMS, and IAG.

The Archive Schema holds the tbArchiveFiles table and the set of 4 archive loading stored procedures.

The AMS, and IAG schemas each hold a set of staging tables used in the Archive Load.

1.4.3 [redacted] ArchiveFiles Table

The table 'Archive.tbArchiveFiles' is used to track and log the export files that are to be loaded. It also logs the progress of each file's loading sequence and notes the success or failure of the process.

Below is the Schema information for the Archive.tbArchiveFiles table with descriptions of each column in the table and any noted configuration values.

Column Name	DESCRIPTION
FullFilePath	Shows the full path and name of each export file that is to be loaded into the Archive database. The column is the Primary Key of the tbArchiveFiles table. This ensures that a given export file can only be loaded once.
iArchiveLoadProcessID	This field is a sequence ID that increments with each batch of files being processed together.
dtPreProcessed	This field contains the date and time the particular file was loaded into the tbArchiveFiles table
vcStageTableName	This field contains the table name for the staging table in the ArchiveStage database that will be first loaded from the export file.
vcFinalTableName	This field contains the table name in the Archive database that will be the final destination for the loading of the export file.
dtStartLoadToStage	Shows the datetime when the particular file started to be loaded to the staging table
dtEndLoadToStage	Shows the datetime when the particular file finished loading to the staging table
bStageLoadState	A bit field to indicate if the file has been successfully loaded to the staging table: 0 = NO 1=YES
vcStageLoadStatus	A message that indicates the status of the loading to the staging table
iRowsLoadedStage	The number of rows loaded to the staging table.
dtStartLoadToFinal	Shows the datetime when the particular file started to be loaded to the final archive table
dtEndLoadToFinal	Shows the datetime when the particular file finished loading to the final archive table

bFinalLoadState	A bit field to indicate if the file has been successfully loaded to the final archive table: 0 = NO 1=YES
vcFinalLoadStatus	A message that indicates the status of the loading to the final archive table
iRowsLoadedFinal	The number of rows loaded to the final archive table. If this value does not match iRowsLoadedStage, then the load is considered to have failed.

1.4.4 ArchiveStage Staging Tables

The AMS and IAG schemas in the ArchiveStage database each hold a set of staging tables used in the Archive Load. These tables have a one to one correlation to the tables being exported from in the Archive Export phase. If a base production table has the name ‘AMS.dbo.tbETCTripTrxn’, then the name of the corresponding staging table will be ‘ArchiveStage.AMS.tbETCTripTrxn’. Note that the staging tables are not exact copies of their corresponding base production tables in all cases. Whenever a view is utilized to export and purge the data for a particular base production table, there will be an ‘extra’ column in the export file that cannot be imported into the final Archive table. The structure of the archive staging tables takes these ‘extra’ columns into account to allow the data to be imported from the export files into the staging tables.

1.4.5 ArchiveStage Stored Procedures

There are 5 stored procedures in the ArchiveStage database that are used to perform the Archive Load process. All five are in the ‘Archive’ schema.

1.4.5.1 [redacted]

This procedure truncates all of the staging tables in the ArchiveStage database. This is the first step of the Archive Load process. Since it does not run until the beginning of the ‘next’ run of the job, the data from the prior run of the Archive Load process remains in the staging tables for any possible checks or troubleshooting.

1.4.5.2 [redacted]

This procedure takes a single parameter (the file path to find the export files to be loaded) and accomplished the following tasks:

1. It pulls the full file path and name for any file found in the file path passed as a parameter
2. It loads these full paths into the ‘FullFilePath’ column of tbArchiveFiles and populates ‘dtPreProcessed’ with the current date and time to show when the filename was loaded in to the table.
3. It assigns the next highest iArchiveLoadProcessID to each file loaded in this batch.
4. It parses the ‘vcStageTableName’ and the ‘vcFinalTableName’ from the name of the export file and sets these values in tbArchiveFiles.
5. The values of bStageLoadState and bFinalLoadState are set to 0 (zero).

6. All other column in tbArchiveFiles are set to NULL.

1.4.5.3 [redacted]

This procedure finds files listed in tbArchiveFiles that need to be loaded to the staging tables and for each table found, it performs the loading to the staging tables as follows:

1. The files that need to be loaded to stage are identified by having bStageLoadState = 0.
2. For each file, a BCP command is created that utilizes the full file path and name as the source and the table name in ‘vcStageTablename’ as the target.
3. Each constructed BCP command is executed in turn, with the start time logged in ‘dtStartLoadToStage’
4. The results of the BCP command are checked to determine 1) if the BCP load was successful, and 2) if it was successful, how many rows were loaded.
5. If the BCP had an error, then the vcStageLoadStatus is set to show the error, bStageLoadState is left as 0, and the dtEndLoadToStage datetime is set.
6. If the BCP was successful, the tne vcStageLoadStatus is set to show the success, the bStageLoadState is set to 1, the iRowsLoadedStage is set, and the dtEndLoadToStage datetime is recorded.

1.4.5.4 [redacted]

This procedure finds files listed in tbArchiveFiles that need to be loaded to the final archive tables and for each table found, it performs the loading to the final archive tables as follows:

1. The files that need to be loaded to stage are identified by having bFinalLoadState = 0.
2. For each file, an Insert / Select statement is created that utilizes the ‘vcStageTableName’ the source and the table name in ‘vcFinalTablename’ as the target. The column list for the insert select is dynamically determined based on the columns found in the final archive table.
3. Each constructed Insert / Select statement is executed in turn, with the start time logged in ‘dtStartLoadToFinal’
4. The results of the Insert / Select are checked to determine 1) if the statement was successful, and 2) if it was successful, how many rows were loaded, and 3) the rows loaded count is compared to the number of rows loaded to the staging table.
5. If the Insert / Select had an error, or the row counts don’t match, then the vcFinalLoadStatus is set to show the error, bFinalLoadState is left as 0, and the dtEndLoadToFinal datetime is set.
6. If the Insert / Select was successful, the vcFinalLoadStatus is set to show the success, the bFinalLoadState is set to 1, the iRowsLoadedFinal is set, and the dtEndLoadToFinal datetime is recorded.
7. In order to be considered a successful load to the final table, the values of iRowsLoadedStage and iRowsLoadedFinal must be equal.

1.4.5.5 [redacted]

After the files for a particular batch have all be successfully loaded to the final Archive tables, this procedure moves the processed files from the ‘Pending’ directory to the ‘Processed Directory’. This clears the ‘Pending’ directory to receive the next month’s batch of export files. The procedure takes the ‘Pending’ and ‘Processed’ paths at its parameters and creates a MOVE statement to move the files. The files are moved only after the procedure confirms that all the files in the latest batch were successfully loaded to stage and final.